文章编号: 1673- 9469(2010) 04- 0106- 07

支持向量机算法的研究及其实现

范玉妹, 郭春静
(北京科技大学 应用科学学院, 北京 100083)

摘要: 支持向量机已成为现在数据挖掘中的研究热点。本文系统地研究分析了 *C*- *SVM*、*v*- *SVM*、*One*- *class SVM* 三种分类算法以及 ε- *SVR*、*v*- *SVR* 两种回归算法, 应用实际数据仿真验证了算法的有效性, 且从参数选择、分类精度、均方误差等方面确定了五种算法各自的优缺点及适用范围。

关键词: 支持向量机; 分类; 回归; Libsvm 软件包; Matlab
中图分类号: O212 文献标识码: A

Study and implement of support vector machine algorithm

FAN Yu-mei, GUO Chun-jing
(School of Applied Science, University of Science and Technology Beijing, Beijing 100083, China)

Abstract: Support vector machine has become the current hot spots in data mining research. This paper researches and analyzes three kinds of classification algorithms such as *C*- *SVM*、*v*- *SVM*、*One*- *class SVM* and the two regression algorithm such as ε- *SVR*、*v*- *SVR*, the paper prove the efficiency of the algorithms through actual data simulation; Advantages, disadvantages and respective applications of the five kinds of algorithms are confirmed from the parameter selection, classification accuracy, and mean square error and so on

Key words: support vector machine; classification; regression; Libsvm software package; Matlab

支持向量机 (support vector machine, SVM) 是数据挖掘中的一项新技术, 是针对小样本数据提出的, 借助于最优化方法解决机器学习问题的新工具。它最初于 20 世纪 90 年代由 Vapnik 提出, 近年来在其理论和算法实现方面都取得了突破性进展, 开始成为克服“维数灾难”和“过学习”等传统困难的有力手段。虽然它还处于飞速发展的阶段, 但是它的理论基础和实现途径的基本框架已经形成。它是建立在统计学习理论基础之上, 对统计学习理论的发展起到巨大的推动作用并被广泛应用。由于其出色的学习性能, 该技术已成为机器学习界的研究热点, 并在很多领域都得到了成功的应用, 如人脸检测、手写体数字识别、文本自动分类等。支持向量机与其他的学习机相比, 具有较好的泛化能力、非线性处理能力和高维处理能力。

1 支持向量分类机算法研究

1.1 *C*- *SVM* 算法

标准的 *C*- *SVM* (*C*- *Support Vector Machines*, 简称 *C*- *SVM*) 算法是样本点线性不可分情况下, 引入松弛变量后的支持向量分类机算法。已知训练向量 $x_i \in R^n$, $i = 1, \cdots, l$ 属于两类, 即 $y_i \in \{1, -1\}$, 把训练数据正确分类是指求得的最优分类超平面不但要把两类数据正确分开, 而且要使分类间隔最大, 引进从输入空间 R^n 到 Hilbert 空间 H 的变换。

$$\Phi: X \subset R^n \rightarrow H$$

$$x \rightarrow X = \Phi(x)$$

把训练集 $T = \{(x_1, y_1), \cdots, (x_l, y_l)\}$

收稿日期: 2010- 09- 03
作者简介: 范玉妹(1948-), 女, 上海人, 教授, 从事概率论与数理统计、运筹学、最优化方向的研究。

映射为 $\overline{T}=\{(x_1, y_1), \cdots, (x_l, y_l)\}=\{(\Phi(x_1), y_1), \cdots, (\Phi(x_l), y_l)\}$。由此得到的初始问题为

$$\begin{cases}\min\limits_{\omega, b, \zeta} = \dfrac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{l}\zeta_i \\ s.t.\ \ y_i[(w\Phi(x_i)) + b] \geq 1-\zeta_i, i=1, 2, \cdots, l \\ \zeta_i \geq 0, i=1, 2, \cdots, l\end{cases}$$

式中 C —惩罚参数, C 越大表示对错误分类的惩罚越大, 它是算法中唯一可以调节的参数; ζ—在训练集线性不可分时引进的松弛因子。

1.2 $v-$ *SVM* 算法

$v-$ *SVM* ($v-$ *Support Vector Machines*, 简称 $v-$ *SVM*), 在 2.1 中讨论的 $C-$ *SVM* 算法中, 有两个相互矛盾的目标: 最大化间隔和最小化训练错误。其中常数 C 起着调和这两个目标的作用。定性地讲, C 值的含义即选取大的 C 值, 意味着更强调最小化训练错误。但定量地讲, C 值本身并没有确切的意义, 所以 C 值的选取比较困难。为此, Scholkoph 提出了 $v-$ *SVM* 模型。所谓 $v-$ *SVM*, 即针对 $C-$ *SVM* 算法中唯一可以调节的参数 C 没有直观解释, 在实际应用中很难选择合适的值, $v-$ *SVM* 中用参数 v 取代 C。在一定条件下, 当样本点个数 $l\rightarrow\infty$ 时, v 以 1 的概率渐近于支持向量个数与样本点个数之比。v 参数可以控制支持向量的数目和误差, 也易选择。

引进从输入空间到希尔伯特空间的变换, 利用这个变换, 由原来的训练集得到新的训练集 $\overline{T}=\{(x_1, y_1), \cdots, (x_l, y_l)\}=\{(\Phi(x_1), y_1), \cdots, (\Phi(x_l), y_l))\}$。

此训练集对应的最优化问题为

$$\begin{cases}\min\limits_{\omega, b, \zeta, \rho}\tau(w, \zeta, \rho) = \dfrac{1}{2}\|w\|^2 - v\rho + \dfrac{1}{l}\sum\limits_{i=1}^{l}\zeta_i \\ s.t.\ \ y_i[(w\Phi(x_i)) + b] \geq \rho - \zeta_i \\ \zeta_i \geq 0, i=1, 2, \cdots, l, \rho \geq 0\end{cases}$$

式中 $\zeta=(\zeta_1, \cdots, \zeta_l)^T$, 这里不含参数 C, 而是换成了参数 v 这个需要实际选定的参数。模型中还多了一个变量 ρ, 我们可以注意到当 $\zeta=0$ 的时候, 约束条件意味着两类点以 $2\rho/\|w\|$ 的间隔被分开。

1.3 *One-class SVM* 算法

One-class SVM 最初是用于高维分布估计, 即用来寻找超平面 VC 维的估计值。对于没有任何分类信息的训练向量 $X_i \in R^n$, $i=1, \cdots, l$, 初始最优化问题为

$$\begin{cases}\min\limits_{\omega, b, \zeta, \rho} \dfrac{1}{2}\|w\|^2 - \rho + \dfrac{1}{vl}\sum\limits_{i=1}^{l}\zeta_i \\ s.t.\ \ w^T\Phi(x_i) + b \geq \rho - \zeta_i \\ \zeta_i \geq 0, i=1, \cdots, l\end{cases}$$

1999 年 Tax 提出用超球面代替超平面来划分数据的想法, 改变了数据集的描述, 目标函数的初始最优化问题变为

$$\begin{cases}\min\limits_{R, \zeta_i, c} R^2 + \dfrac{1}{vl}\sum\limits_{i=1}^{l}\zeta_i \\ s.t.\ \ \|\Phi(x_i) - c\|^2 \leq R^2 - \zeta_i \\ \zeta_i \geq 0, i=1, \cdots, l\end{cases}$$

其中 R 为超球面的半径, 通过设定参数 $0 \leq v \leq 1$, 使超球面的半径和它所能包含的训练样本数目之间进行折衷。当 v 小的时候, 尽量把数据放进球里面, 当 v 大的时候, 尽量压缩球的尺寸, 使用 Lagrangian 函数来解这个优化问题。

2 支持向量回归机算法的研究

2.1 $\varepsilon-$ *SVR* 算法

$\varepsilon-$ *SVR* 采用了 $\varepsilon-$ 不敏感损失函数, 所以它有一个明显的优点即具有稀疏性: 所有在 $\varepsilon-$ 带内部的样本点都不是支持向量, 它们对决策函数没有贡献。换句话说, 去掉这些样本点, 不会影响最终的决策函数, 只有支持向量才会影响决策函数。

$\varepsilon-$ *SVR* 的原始问题如下:

$$\begin{cases}\min\limits_{\omega, \zeta^{(*)}, b} \tau(w, \zeta^{(*)}) = \dfrac{1}{2}\|w\|^2 + C\cdot\dfrac{1}{l}\sum\limits_{i=1}^{l}(\zeta_i + \zeta_i^*) \\ s.t.\ \ ((w\cdot x_i) + b) - y_i \leq \varepsilon + \zeta_i, i=1, \cdots, l \\ y_i - ((w\cdot x_i) + b) \leq \varepsilon + \zeta_i^*, i=1, \cdots, l \\ \zeta_i^{(*)} \geq 0, i=1, \cdots, l\end{cases}$$

2.2 $v-$ *SVR* 算法

在 $\varepsilon-$ *SVR* 中, 需要事先确定 $\varepsilon-$ 不敏感损失函数中的参数 ε。然而在某些情况下选择合适的 ε 并不是一件容易的事情。因此, 与 2.2 中的分类算法类似, 引进参数 v 取代 ε, 构造能够自动计算 ε 的一种变形算法—$v-$ *SVR*。引进参数 v 后, 则可将 3.1 中的最优化问题修改为如下 $v-$ *SVR* 问题:

$$\begin{cases}\min\limits_{\zeta^{(*)}, \varepsilon, b} \tau(w, \zeta^{(*)}, \varepsilon) = \dfrac{1}{2}\|w\|^2 + C\cdot(v\varepsilon + \dfrac{1}{l}\sum\limits_{i=1}^{l} \\ (\zeta_i + \zeta_i^*) \\ s.t.\ \ ((w\cdot x_i) + b) - y_i \leq \varepsilon + \zeta_i, i=1, \cdots, l \\ y_i - ((w\cdot x_i) + b) \leq \varepsilon + \zeta_i^*, i=1, \cdots, l \\ \zeta_i^{(*)} \geq 0, \varepsilon \geq 0 \quad i=1, \cdots, l\end{cases}$$

3 仿真实例及结果分析

仿真实例是利用 matlab 软件和 Libsvm2.89 版本来实现的, 在实验中所用到的核函数类型均为径向基核函数, 分类实验数据来源均为 UCI 学习库, 回归实验数据来源为公司实际数据。

3.1 分类实例及结果

本文采用一个大样本与一个小样本数据分别用三种不同的算法对其进行训练与预测。其中, 大样本数据集有 83 个特征, 训练集 train1 包含 1 605 个数据, 测试集 test1 包含 30 956 个数据; 小样本数据集只有两个特征, 训练集 train2 包含 500 个样本点, 测试集 test2 包含 362 个样本点。

从表 1 我们可以看到采用 *C*- *SVC* 算法对数据 train1 和 test1 大样本数据进行训练和测试时, 参数对(*C*, γ) 的变化对实验结果的影响, 最重要的结果是对训练集和测试集的分类正确率。正确率越大, 说明分类准确度越高, 算法越适用于这个数据集。从表 1 可看出, 此算法对 train1 和 test1 的分类正确率只能达到 80% 左右, 相对比较低。此算法中, 当支持向量个数越多时, 对训练集和测试集的分类正确率都越低, 这个结果也是符合前面理论部分的, 而且, 我们看到对训练集的正确率越高, 相应的对测试集的测试正确率也越高, 这说明我们做十折交叉验证实验得到最优参数对, 再选择最优参数对进行训练得到训练模型, 最后用测试集进行测试是很有必要的。从表 1 整体来看, 用 *C*- *SVC* 算法对 train1 数据集进行训练支持向量的个数很多, 正确率不高, 对测试集 test1 的分类正确率也不高, 说明此算法对所选的大样本数据是不太适用的。

表 2 反应的是采用 *v*- *svm* 算法对数据 train1 和 test1 大样本数据进行训练和测试时, 参数对(*v*, γ) 的变化对实验结果的影响。其中, 参数对的变化对训练集及测试集分类正确率的影响不太明显, 一般在 80% 左右波动。从表 2 还可看到, 当选

表 1 *C*- *SVM* 算法对 train1 和 test1 的训练和测试结果

Tab.1 The training and testing results for train1 and test1 with *C*- *SVM* algorithm

(*C*, γ)	迭代次数	支持向量个数及所占比率	边界支持向量个数及所占比率	训练集分类正确率	测试集正确分类个数及正确率
(10, 0.1)	2 633	1 328(82.7%)	25(1.56%)	79.514 3%	77.274 2% (23 921/30 956)
(100, 0.1)	2 633	1 328(82.7%)	25(1.56%)	79.514 3%	77.274 2% (23 921/30 956)
(10, 0.2)	2 908	1 574(98.0%)	24(1.50%)	76.463 3%	75.946 5% (23 510/30 956)
(10, 0.01)	3 542	683(42.6%)	374(23.3%)	82.939%	83.263 3% (25 775/30 956)
(2, 0.2)	2 901	1 574(98.0%)	29(1.8%)	76.463 3%	75.946 5% (23 510/30 956)

表 2 *v*- *SVM* 算法对 train1 和 test1 的训练和测试结果

Tab.2 The training and testing results for train1 and test1 with *v*- *SVM* algorithm

(*v*, γ)	迭代次数	支持向量个数及所占比率	边界支持向量个数及所占比率	训练集分类正确率	测试集正确分类个数及正确率
(0.1, 0.01)	6 437	657(40.934 5%)	49(3.053 0%)	79.950 2%	79.719 6% (24 678/30 956)
(0.2, 0.01)	4 830	669(41.495 3%)	152(9.470 4)	81.755 9%	80.494 9% (24 918/30 956)
(0.3, 0.01)	2 383	681((42.429 9%)	346(21.557 6%)	82.814 4%	82.940 3% (25 675/30 956)
(0.01, 0.01)	2 340	578(36.012 5%)	0(0%)	78.953 9%	79.580 7% (24 635/30 956)
(0.1, 0.1)	2 323	1 325(82.554 5%)	23(1.433 0%)	79.514 3%	77.274 2% (23 921/30 956)
(0.2, 0.1)	2 382	1 331(82.928 3%)	40(2.492 2%)	79.638 9%	77.406 6% (23 962/30 956)
(0.01, 0.1)	973	169(10.529 5%)	9(0.560 7%)	57.036 1%	30.042 6% (9 300/30 956)

择参数不适当时, 得到的正确率是很低的, 如当 v = 0.01, γ= 0.1 时, 对训练集及测试集的正确率分别只有 57% 和 30%, 这个结果是很差的。因此, 根据特定的数据集选择合适的参数对进行训练是很有必要的。用 $v-svm$ 算法对 train1 数据集进行训练支持向量的个数较多, 正确率不高, 对测试集 test1 的分类正确率也不高, 说明此算法对所选的大样本数据是不太适用的。

从表 3 我们可以看到, 此表中的结果与表 1、表 2 的结果不太一样, 主要反应在对训练集和测试集的分类正确率上。如当 γ 都取 0.1, v 分别取 0.01, 0.1, 0.9 时, 对训练集的分类正确率分别为 55.977 6%、56.164 4%、72.602 7%, 有较大的变化, 但对测试集的分类正确率都为 75.946 5%, 没有变化, 这个结果跟其它两个算法是不一样的。当 γ 都取 0.01, v 分别取 0.01, 0.1, 0.9 时, 对训练集的分类正确率分别为 26.525 5%、29.078 5%、71.731%, 最低为 26%, 这个正确率是相当低的, 而对测试集的分类正确率确有 75.365% 变化不大。整体来看, 此算法对 train1 和 test1 数据集训练和测试的正确率不高, 最低有 20% 多, 最高不到 76%, 比前面两个算法的精度都差。说明此算法对这个数据集很不适用。另外, 也反应了参数选择的重要性, 如果参数选择不当, 则最后得到的结果尤其是正确率会特别不理想。

通过表 1、表 2 与表 3 之间的对比, 三个表反应的是三种算法对大样本数据集的训练和预测结果, 参数的变化对结果都有不同程度的影响, 说明我们在对具体的数据集进行训练和预测时选择适当的参数的必要性。另外, 通过比较发现, 采用 $C-svm$ 与 $v-svm$ 算法, 正确率最高能达到 83% 左右, 而采用 $one-class\ svm$ 算法正确率最高不到 76%, 说明对具体的数据集选择适当的算法的必要性, 也说明了改进的算法不一定适合于所有的数据集。从以上分析看, 这三种算法的训练和测试正确率并不高, 说明它们不太适用于大样本数据集。表 4 采用 $C-svm$ 算法对小样本数据集进行训练和测试时, 参数对 (C, γ) 的变化对实验结果的影响。从正确率方面看, 算法的结果比较理想, 对训练集的正确率最高达到了 99.6% 接近 100%, 对测试集的正确率也达到了 99% 以上, 此算法对这样的小样本数据的扩展性很好, 但当参数取得不合适时, 如 γ= 0.9, C= 1 时, 正确率只有

表 3 $one-class\ svm$ 算法对 train1 和 test1 的训练与测试结果

Tab. 3 The training and testing results for train1 and test1 with *one-class svm* algorithm

(v, γ)	迭代次数	支持向量个数及所占比率	边界支持向量个数及所占比率	训练集分类正确率	测试集正确分类个数及正确率
(0.01, 0.1)	1 260	939(58.5%)	0(0%)	55.977 6%	75.946 5% (23 510/30 956)
(0.1, 0.1)	1 655	949(59.1%)	0(0%)	56.164 4%	75.946 5% (23 510/30 956)
(0.9, 0.1)	230	1 478(92.1%)	1 420(88.5%)	72.602 7%	75.946 5% (23 510/30 956)
(0.01, 0.01)	142	60(3.7%)	1(0.06%)	26.525 5%	75.365% (23 330/30 956)
(0.1, 0.01)	250	182(11.3%)	141(8.8%)	29.078 5%	75.468 4% (23 362/30 956)
(0.9, 0.01)	165	1 447(90.2%)	1 444(90.0%)	71.731%	75.946 5% (23 510/30 956)
(0.9, 1)	3 960	1 596(99.4%)	0(0%)	74.782 1%	75.946 5% (23 510/30 956)

表 4 $C-SVM$ 算法对 train2 和 test2 的训练和测试结果

Tab. 4 The training and testing results for train2 and test2 with *C-svm* algorithm

(C, γ)	迭代次数	支持向量个数及所占比率	边界支持向量个数及所占比率	训练集分类正确率	测试集正确分类个数及正确率
(200, 0.9)	5 137	55(11%)	41(8.2%)	99.6%	99.171 3% (359/362)
(100, 0.9)	2 201	73(14.6%)	60(12%)	99.4%	98.895% (358/362)
(10, 0.9)	487	165(33%)	154(30.8%)	94.4%	89.502 8% (324/362)
(1, 0.9)	180	215(43%)	208(41.6%)	82.2%	79.005 5% (286/362)
(1, 0.01)	178	329(65.8%)	327(65.4%)	68.6%	64.917 1% (235/362)

80% 多, 再一次说明了参数选择的重要性。从表 4 来看, 用 *C* - *SVM* 算法对训练集 train2 数据集进行训练支持向量的个数不多, 能达到的最高正确率很高, 对测试集 test2 能达到的分类正确率也很高, 说明此算法对所选的小样本数据是很适用的。

表 5 反应的是采用 *v* - *svm* 算法对数据集 train2 和 test2 小样本数据进行训练和测试时, 参数 (v, γ) 的变化对实验结果的影响。从分类正确率上来看, 参数的变化对结果有明显的影响, 对训练集和测试集的最高正确率都达到了 99% 以上, 而最低分别为 50% 左右和 47% 左右。从表 5 同样可看到, 当选择参数不适当时, 得到的正确率是很低的, 如当 v= 0.05, γ= 0.1 时, 对训练集及测试集的正确率分别只有 38% 和 51%, 这个结果是很差的。因此, 根据特定的数据集选择合适的参数对即进行模型选择进行训练是很有必要的。从表 5 来看, 用 *v* - *svm* 算法对 train2 数据集进行训练支持向量的个数不多, 能达到的最高正确率很高, 对测试集 test2 能达到的分类正确率也很高, 说明此算法对所选的小样本数据是比较适用的。

表 6 的结果与表 4、表 5 的结果不太一样。主要反应在对训练集和测试集的分类正确率上。从表中结果我们看到, 采用 *one* - *class svm* 算法对 train2 训练集的正确率最高只能达到 67% 左右, 对 test2 测试集的正确率最高也只能达到 60% 左右, 这样的结果是很不理想的。从表 5 可以看到参数的变化并不能提高正确率, 说明此算法对所选取的数据集是不适用的。

通过表 4、表 5 与表 6 之间的对比, 参数的变化对结果都有不同程度的影响, 说明我们在对具体的数据集进行训练和预测时选择适当参数的必要性。另外, 通过比较发现, 采用 *C* - *svm* 与 *v* - *svm* 算法, 正确率最高能达到 99% 以上, 而采用 *one* - *class svm* 算法正确率最高不到 70%, 说明对

表 5 *v* - *SVM* 算法对 train2 和 test2 的训练和测试结果

Tab. 5 The training and testing results for train2 and test2 with *v* - *svm* algorithm

(v, γ)	迭代次数	支持向量个数及所占比率	边界支持向量个数及所占比率	训练集分类正确率	测试集正确分类个数及正确率
(0.1, .0.9)	1 771	61(12.2%)	41(8.2%)	99.6%	99.171 3% (359/362)
(0.1, 0.001)	23	51(10.2%)	49(9.8%)	50.8%	48.066 3% (174/362)
(0.1, 0.01)	64	66(13.2%)	66(13.2%)	58.4%	46.685 1% (169/362)
(0.05, 0.8)	991	38(7.6%)	11(2.2%)	99.4%	99.447 5% (360/362)
(0.05, 0.7)	1 231	42(8.4%)	10(2.0%)	99.4%	99.171 3% (359/362)
(0.05, 0.1)	147	64(12.8%)	64(12.8%)	38%	51.657 5% (187/362)
(0.005, 0.8)	162	38(7.6%)	0(0%)	93.4%	92.817 7% (336/362)

表 6 *one* - *class svm* 算法对 train2 和 test2 的训练和测试结果

Tab. 6 The training and testing results for train2 and test2 with *one* - *class svm* algorithm

(v, γ)	迭代次数	支持向量个数及所占比率	边界支持向量个数及所占比率	训练集分类正确率	测试集正确分类个数及正确率
(0.1, 0.9)	73	53(3.30%)	45(2.80%)	38%	44.475 1% (161/362)
(0.01, 0.9)	87	10(0.62%)	3(0.19%)	36.4%	39.502 8% (143/362)
(0.001, 0.9)	22	10(0.62%)	0(0%)	36.6%	39.502 8% (143/362)
(0.1, 0.1)	53	52(3.23%)	49(3.05%)	36.4%	43.646 4% (158/362)
(0.1, 0.01)	54	52(3.23%)	49(3.05%)	36.4%	45.027 6% (163/362)
(0.01, 0.000 1)	1	5(0.31%)	5(0.31%)	57%	56.906 1% (206/362)
(0.001, 0.000 1)	0	1(0.06%)	0(0%)	67.2%	60.497 2% (219/362)

具体的数据集选择适当的算法的必要性, 也说明了改进的算法不一定适合于所有的数据集。*C*- *svm* 与*v*- *svm* 的训练和测试正确率都能达到理想的结果, 说明它们适用于这样的小样本数据集。一般, 在实际问题中要具体问题具体分析, 尽量选择合适的参数对, 使得正确率尽量高, 提高算法的精度。然而, 我们可以看到, 对 *C*- *svm* 算法而言, *C* 的变化范围很大, 对具体实力来说是比较难选取合适的 *C* 值的, *v*- *svm* 作为*C*- *svm* 算法的改进算法, 用参数 *v* 代替参数*C*, *v* 的选取相对较容易且在算法迭代过程中, *C* 值作为结果的一部分自动计算。用两种算法对小样本数据集的训练结果正确率都能达到理想的效果, 精确度都很高。

最后, 比较同一种算法对不同的训练集和测试集的结果。通过表 1 与表 4 的结果我们不难发现, 采用 *C*- *svm* 算法不管对大样本数据集 train1、test1 还是对小样本数据 train2、test2 来说, 选择不同的参数对对实验结果都有不同程度的影响, 而从分类正确率的角度来看, 对大样本数据集的最高正确率达到 83% 左右, 对小样本数据的最高正确率能达到 99% 以上, 接近 100%, 这充分说明了此算法较适用于小样本数据。同样, 通过表 2 与表 5 的对比发现 *v*- *svm* 算法较适用于小样本数据集, 只是它在参数选择上更容易一些。通过表 3 与表 5 的比较发现, *one*- *class svm* 算法对本文选取的大样本与小样本数据集的分类正确率都不理想, 即它对一般的数据集并不适用, 还不如最基本的 *C*- *svm* 与*v*- *svm* 算法效果好。

3.2 回归实例及结果

此部分本文选用公司实际数据, 数据集 train3 有 3 个特征, 共有 130 个样本点。

表 7 反应的是采用 ε- *SVR* 算法对数据集进行训练时, 参数变化对实验结果的影响。最重要的结果是均方误差和平方相关系数, 其中均方误差越小, 说明回归的准确度也越高, 即拟合曲线与实际的越接近, 平方相关系数的大小是对变量与特征之间的相关程度的一种度量, 其值越大即越接近 1, 表示变量与特征之间相关程度也越大。从表 8 的结果可以看到, γ 一定 *C* 变化时, 对均方误差和平方相关系数没有影响。均方误差的值很小, 都为 0.004 054 63, 说明此算法对 train4 的训练结果准确度较高。平方相关系数的值都为 0.976 418, 比较接近 1, 说明变量与三个变量间的相关程度很高。当均方误差越小时, 平方相关系数相对变大, 这也是符合实际情况的, 当均方误差越小时, 说明算法回归的准确度越高, 自然变量与特征之间的平方相关系数越大。

从表 8 的结果可以看出 *v* 一定 γ 变化时, 对均方误差的影响并不明显, 其值基本在 0.001 左右波动, 但也有特殊情况, 如 γ= 0.01 时, 均方误差达到了 0.018, γ 对平方相关系数的影响也不太明显, 其值都在 0.9 以上, 但整体来说平方相关系数的值很接近 1, 说明变量与特征之间的相关程度很

表 7 ε- *SVR* 算法对 train4 的训练结果

Tab. 7 The training results for train4 with ε- *SVR* algorithm

(*C*, γ)	迭代次数	支持向量个数及所占比率	均方误差	平方相关系数
(1, 0.5)	12	6(4.6%)	0.004 054 63	0.976 418
(10, 0.5)	12	6(4.6%)	0.004 054 63	0.976 418
(100, 0.5)	12	6(4.6%)	0.004 054 63	0.976 418
(100, 0.1)	13	3(2.3%)	0.003 384 47	0.975 456
(100, 0.01)	16	3(2.3%)	0.003 142 38	0.962 77
(100, 0.001)	20	5(3.8%)	0.003 210 96	0.960 822

表 8 *v*- *SVR* 算法对 train4 的训练结果

Tab. 8 The training results for train4 with *v*- *svr* algorithm

(*C*, γ)	迭代次数	支持向量个数及所占比率	均方误差	平方相关系数
(0.1, 0.9)	327	18(13.5%)	0.001 433 52	0.978 447
(0.1, 0.1)	59	13(10.0%)	0.001 200 07	0.978 201
(0.1, 0.01)	22	12(9.2%)	0.017 901 3	0.933 88
(0.3, 0.1)	263	36(27.7%)	0.000 936 021	0.983 026
(0.5, 0.1)	375	58(44.6%)	0.000 915 293	0.983 34
(0.01, 0.1)	10	5(3.8%)	0.021 848 8	0.963 626

高。对均方误差、平方相关系数结果无明显影响。从以上分析可以看出,此算法对 train4 数据集的训练结果是比较稳定的,无明显偏差。

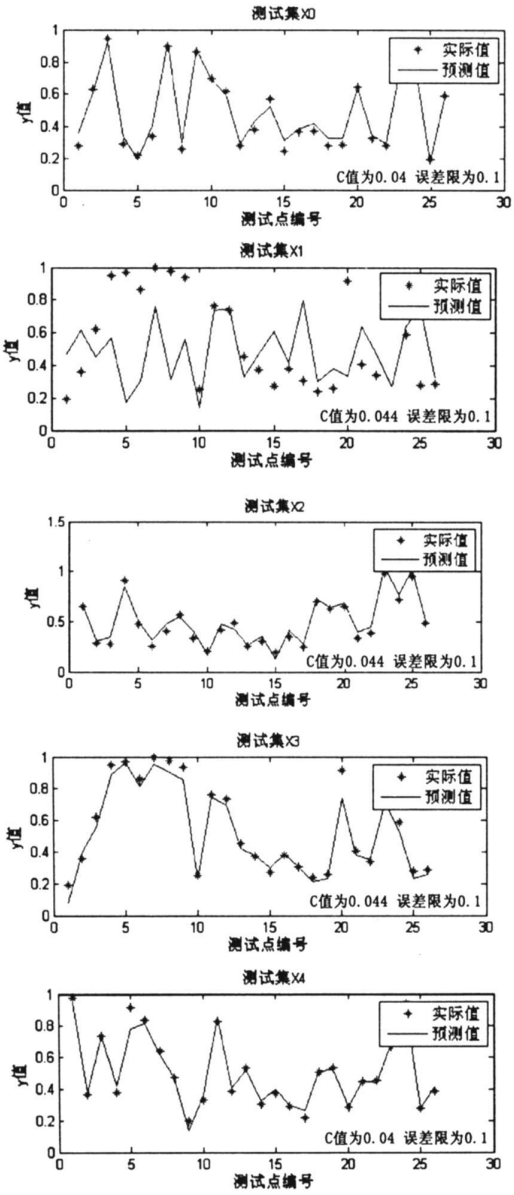

图1 对train4数据集做5折交叉验证实验的拟合结果图

Fig.1 The fitting results for doing five fold cross validation experiment of the data set

从表7与表8的结果对比发现,无论是 ε- *svr* 还是 *v* - *svr* 算法,选取不同的参数对对实验结果都有不同程度的影响。因此,在实际问题中要具体问题具体分析,尽量选择合适的参数对,使得均方误差尽量小,提高算法的精确度。对 ε- *svr* 算法而言,*C* 的变化范围很大,对具体实例来说是比较难选取合适的 *C* 值的,作为 ε- *svr* 算法的改进算法,用参数 *v* 代替参数 *C*,*v* 的选取相对较容易且在算法迭代过程中,*C* 值作为结果的一部分自动计算。用两种算法对 train4 这个小样本数据集的训练结果均方误差都很小,精确度都很高,而且当参数变化时,对均方误差的大小无明显影响,说明算法都比较稳定。

用 Matlab 实现 ε- *SVR* 算法,对 train4 数据集做5折交叉验证实验的拟合结果如图1所示。

4 结论

1) 标准的支持向量机 *C*- *SVM* 是一个基础算法,可以对所有的数据进行分类。但是,从正确率来看,比较适用于小样本数据集。

2) *v* - *SVM* 中支持向量的取值可以通过选择合适的 *v* 来得到一个好的结果,但是对于大样本问题还是需要核函数中参数的调节才能得到比较满意的结果。

3) *One*- *class SVM* 对一般数据集的分类正确率并不高,所以不适用于一般的数据集。

4) 对于所选取的小样本数据集,两种回归算法的结果都是比较理想的,即分类均方误差较小,平方相关系数都接近1,且结果随参数对的变化是不明显的,算法比较稳定。

5) 不论分类算法还是回归算法,参数的变化对实验结果都有不同程度的影响。

参考文献:

[1] 邓乃杨,田英杰. 数据挖掘中的新方法- 支持向量机[M]. 北京: 科学出版社, 2004.

[2] 云 潜,张学工. 支持向量机函数拟合在分形插值中的应用[J]. 清华大学学报(自然科学版), 2000, 40(3): 76- 78.

[3] 刘华富. 一种快速支持向量机分类算法[J]. 长沙大学学报, 2004, 17(7): 40- 41.

[4] VAPNIK N. The nature of statistical learning theory[M]. New York: Springer- Verlag, 1995.

[5] REYZIN L, SCHAPIRE R E. How boosting the margin can also boost classifier complexity[C]// ICML' 06, the 23rd Int. Conf., on Machine Learning Pittsburgh, Pennsylvania, USA, 2006: 753- 760.

[6] PLATT J. Fast training of support vector machines using sequential minimal optimization[C]// in Advances in Kernel Methods - - Support Vector Learning, B. Schoelkopf, C. Burges, and A. Smola Eds., 1999: 185- 208.

[7] VAPNIK V. Statistical learning theory[M]. New York: John Wiley&Sons, 1998.

[8] COTTES C, VAPNIK V N. Support vector networks[M]. Mach Learn, 1995.

[9] YU H, YANG J, HAN J. Classifying large data sets using SVMs with hierarchical clusters[C]// in proc. of the ACM SIGKDD Int. Conf. on KDD, 2003: 306- 315.

(责任编辑 刘存英)